Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 68 (2015) 42 – 52

Procedia Computer Science

HOLACONF - Cloud Forward: From Distributed to Complete Computing

CIRANO: An integrated programming environment for multi-tier cloud based applications

George Fylaktopoulos[a], Georgios Goumas[b], Michael Skolarikis[a], Aris Sotiropoulos[c], Dimitrios Athanasiadis[a], Ilias Maglogiannis[d]*

[a]*B-Open S.A., Laskaratou 11A Thessaloniki Pylaia 54250*
[b]*Computing Systems Laboratory, National Technical University of Athens, Greece*
[c]*GRNET S.A., Av. Mesogion 56, 11527, Athens, Greece*
[b]*Department of Digital Systems, University of Piraeus, Greece*

Abstract

This paper describes the CIRANO platform, a cloud Integrated Development Environments (IDE) that substantially supports Model Driven Development (MDD) and team collaboration, in order to facilitate the development of cloud–based applications. The paper presents the state of the art in the field and discusses the technical details of CIRANO architecture and its modular implementation. The main features of the proposed platform are presented as a case study application concerning the update and porting of an existing collaboration system, called Comidor. The paper discusses the findings in comparison with existing tools and proposes extensions of the platform as future work.

Peer-review under responsibility of Institute of Communication and Computer Systems.

Keywords: Cloud; Integrated Development Environments; collaboration systems, programming environments

1. Introduction

The global penetration of the Internet, along with the affordability of Internet devices (laptops, tablets, smart phones, game consoles), formed the need for IT mobility that led to the emergence of Cloud[1]. The Cloud has flourished quickly and plenty of platforms for Cloud software development, as well as Cloud programming environments, are now available in the market. In this context, a large number of applications from various domains

* Corresponding author: Ilias Maglogiannis University of Piraeus, Grigoriou Lampraki 126 Piraeus Greece Tel.: +30-2104142517; fax: +30-2104142517. *E-mail address:* imaglo@unipi.gr

Peer-review under responsibility of Institute of Communication and Computer Systems.
doi:10.1016/j.procs.2015.09.222

including file hosting, social networking, office applications, business applications, games and numerous others have been ported from desktops or servers to cloud infrastructures. This provides clear benefits to users, who are released from the burden to install, configure and maintain applications, as well as hosting hardware[2]. This trend has created an increasing need for fast and cost-effective cloud application development. Not surprisingly, the software development environment itself, which incorporates a wide tool-chain of editors, compilers, runtime environments, debugging and documentation tools etc., has also found its way to cloud infrastructures, which can offer shared programming and computing resources, together with powerful collaboration and code sharing facilities[3].

During the last few years, a large number of cloud Integrated Development Environments (cloud IDEs) have emerged, including Compilr (`https://compilr.com`), JS-Fiddle (`https://jsfiddle.net`), Cloud9 (`https://c9.io`), AkShell (`https://github.com/akshell`), Codenvy (`https://codenvy.com`), Eclipse Orion (`http://www.eclipse. org/orion`), Coderun (`www.coderun.com`), Kodingen (`https://koding.com`) and others. Typically, these platforms support code development with the aid of an enhanced web-based editor, while code compilation, execution and testing is performed on remote cloud infrastructures. In several cases, code productivity is enhanced by some sort of code reuse through shared user repositories, while some platforms also provide a cloud hosting environment for post-production application execution. However, the intense need for rapid and cost effective application development, poses even tougher requirements to the process of application development. Developer collaboration and software reuse need to reach their maximum, in order to deliver high quality applications with reduced time to market.

Model Driven Development (MDD)[4,5,6] is a successful software engineering process in traditional application development processes. MDD decomposes system design and operational logic from implementation details, by utilizing appropriate abstractions expressed as models. This decomposition greatly simplifies the software development process and is able to automate substantial parts of it, by exploiting automatic mechanisms for code generation. Despite the fact that typical cloud applications share a large number of common features and components, MDD is hardly embraced by modern cloud IDEs.

In this paper we present CIRANO, a cloud IDE that substantially supports MDD to facilitate the development of cloud applications. Beyond common functionality existing in modern IDEs like enhanced editors, debugging tools and support for popular databases, CIRANO supports a) component-based application development, b) multilayer programming, which greatly facilitates the online collaboration between developers, c) automatic database development and connectivity and d) an internal model and functionality repository.

The rest of the paper is organized as follows: In Section 2 we provide an overview of the existing Cloud Based Development Solutions. In Section 3 we describe the technical details of the proposed CIRANO platform, while in Section 4 we discuss its operation in practice. Finally, Section 5 concludes the paper.

2. Related Work and Background Information

Cloud based development refers to the complete transfer of developer workspace into the cloud. The developer's environment is a combination of the programming IDE, the local build system, the local runtime, the connections between these components and their dependencies. The cloud-based workspace is centralized, making it easy to share. Developers can invite others into their workspace to co-develop, co-edit, co-build, or co-debug SW. Thus, the existing Cloud Based Development Solutions cover the broad spectrum of the above operations. A typology of these tools is as follows:

Cloud programming environments. These are online web based platforms designed to offer development capabilities to developers. They usually consist of a source code editor, a number of compilers or interpreters depending on the programming language, a debugger and a project/solution viewer for managing the independent subcomponents. Apart from the default features and due to high competition, programming environments have evolved often including connections to code repositories, collaboration features for code sharing, Virtual Machines (VMs) for instant application deployment and monitoring tools.

Cloud Repositories, which are web hosting facilities that leverage the strengths of well-known version control systems such as Git (`https://git-scm.com`), Subversion (`https://subversion.apache.org`) and Mercurial (`https://mercurial.selenic.com`). They often support services and tools, including bug tracking, release management, mailing lists, and wiki-based documentation.

Cloud SW Modeling tools, which refer to applications used to describe the functional and non-functional requirements of a project. The specific tools usually include a designer for presenting the architecture of an information system, a process and an interface or a component. The most well-known language used by these tools is UML (Unified Modelling Language)[7]. Most of the desktop IDEs support plugins for UML diagrams (like Rational Rose and BOUML), however there are open source tools like Argo UML (`argouml.tigris.org`) and StarUML (`staruml.io`), which have been widely used for desktop applications. Nevertheless, in the Cloud era the ability to create UML diagrams has been expanded through cloud modeling services, which are able to create any kind of diagrams.

Cloud SW Composition Tools, which refer to complete development environments covering all phases of applications coding lifecycle. These phases include editing, compiling, debugging, linking, testing and maintaining the application's source code. These tools are often referred to as Integrated Development Environments–IDEs and have enjoyed great acceptance in desktop applications, because of their user friendly interfaces and intelligent code management (live syntax debugging, code optimization, ready UI components and libraries). Some of the most commonly used desktop IDEs are: Visual Studio (`https://www.visualstudio.com`), Eclipse (`https://eclipse.org`) and Netbeans (`https://netbeans.org`).

Cloud SW processing and documentation tools cover the need of having an integrated hub of help and reference information. The variety of software within organizations make it difficult, if not impossible, to handle them efficiently. These tools provide advanced search filters, offer a single knowledge database pool and also enhance collaboration within teams.

Finally, elasticity[8] and scalability[9] are two of the main advantages of the Cloud Technology. However, flexibility in resources depending on live demand is an operation that requires constant monitoring of the executed Cloud applications. For this reason, **Cloud management and orchestration tools** have been developed. The table below indicates the main Cloud IT operational areas, along with Cloud software tools for each category.

Table 1. Existing cloud platforms.

Programming Environments	• Compilr • jsFiddle • JavaWIDE • Cloud9 • Akshell • Codenvy • Orion • Coderun • Koding • Codeanywhere	Repositories	• Bitbucket • ProjectLocker • CloudForge • GitHub • SourceForge • LaunchPad • Assembla • CodePlex • Project Hosting • GitLab
Cloud SW Composition Tools	• Visual Studio • Eclipse • Netbeans	Cloud SW processing and documentation tools	• ClickHelp • Helpiq • Robohelp • HelpServer
Cloud SW Modelling	• Genmymodel • visual-paradigm • Creately • Diagram • Gliffy	Cloud SW management and orchestration tools	• Azure Preview Portal • Rackspace Cloud Monitoring • vRealize • Hyperic • Ganglia • Nagios • Zabbix

Cloud is a dynamic environment that follows the fast changes of technology's progress and trends, resulting in several of the aforementioned tools failing to keep up. For example, the programming environment tools (Akshell (`https://github.com/akshell`) and JavaWIDE[10], that focused strictly on offering programming environments without any deployment options, soon went out of business. Moreover, current solutions provide restricted support

for modelling, which are not fully incorporated in a development environment to provide an end-to-end MDD-based development chain.

3. The proposed CIRANO platform

In this section we present CIRANO, a novel cloud-based IDE that supports MDE in a unified way. As illustrated in the overall architecture (Figure 1), CIRANO consists of the following fundamental units:

- The Application Server, which is responsible for the database connectivity, manages the business logic, accepts and controls third party connections and services, binds system resources using pooling techniques. It also manages the deployed programing units and controls the user authorization on them.
- The Multi-tenancy module, which provides reusability of shared resources among users, like database metadata, and is a necessity for cloud environments.
- The Presentation Manager, which is responsible for the generation of user interfaces.
- The Application Session Manager, which manages functionalities and data stored until the user session has expired.
- The IDE Module, which is used by development teams to create applications.
- The Applications folder, where all application files are stored.
- An Auditing Server, which provides real time user action auditing.
- A Chat Server, offering real time communication capabilities.
- A Billing Server, used for internal charging of clients.
- A Messenger Server, responsible for internal and external (through a Mail Server) message exchanging.
- A Database Server, depending on the installation and the user application needs

Developers use a web browser to access the IDE Module in order to design and implement a Cloud application. Requests are passed through the Application Session Manager to the Application Server, which gathers data from the Database Server and business logic from the Applications. It then transforms this information into a user interface using the Presentation Manager and sends it back to the user's browser. Shared data among different users reside in the Multi-Tenancy Module. All Servers can connect to the Application Server to access database and application information.

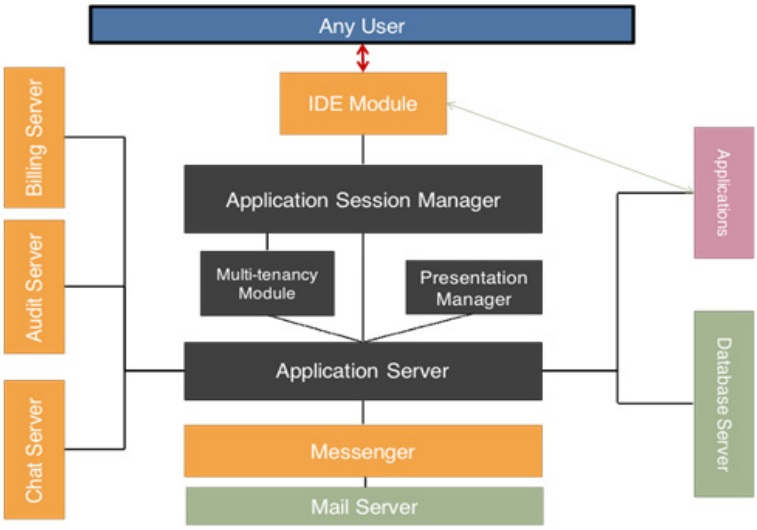


Fig. 1. Overall CIRANO architecture

CIRANO platform follows the Model-Driven Architecture (MDA)[6], defining three types of models for the representation of the functional and nonfunctional behavior of the system (see Figure 2):

1. The Computational Independent Model (CIM), which is an abstract layer that describes the behavior of the application, without revealing any technical details.
2. The Platform Independent Model (PIM), which describes the system and its subsystems in a platform independent way, keeping the system free from installation limitations.
3. The Platform Specific Model (PSM), which extends the PIM, showing the details required for the system's installation on a certain platform.

Each of the aforementioned layers includes a number of available components. More specifically, CIRANO CIM (**CCIM**) includes the code model, which consists of the representation of the application's business logic written in PlatonScript, the platform's custom programming script language. Code is based on events (database or user–based events), the data model, which represents the unit's data in a typical ER diagram and the page model, which defines the necessary UI for the users interaction with the data and the code model.

Fig. 2. Platform components per layer

CPIM defines the real time components that are dynamically created during runtime by the Application Server. These components include:

- Data objects: They can be any kind of data collections which are required by the unit's specific functionalities.
- Filter objects: They are criteria and selectors which act on the data objects in order to filter their data.
- Code objects: Procedures written in PlatonScript.
- User Interface Objects: Graphical components (like tables, forms, fields, graphs) which are connected with the data objects and presented to the final user.
- Menu Objects: They specify the user access to application's units.
- Document Objects: Its unit's documentation is included in these objects.

Finally, **CPSM** contains the platform specific components used during CIRANO's deployment. They include the VMs, the OS–specific Java Runtime Environment, the appropriate database instances and the required mail servers (if they are available).

In cloud environments, multi-tenancy and isolation are crucial for the ability of the system to share and reuse resources, as well as provide security on each application's data. One of the most well-known methods of providing isolation is the Virtual Multi-tenancy, which is based on the creation of a virtual resource. However, the coexistence of many VMs under the same natural resource has been proved ineffective, since their interaction affects the system itself. CIRANO's architecture is based on an alternative isolation technique, known as Organic Multi-tenancy, which relies on different layers of the PaaS platform, such as the application server layer or the DBMS layer.

The structure depicted in Figure 3 is an example platform specific model of CIRANO's installation and deployment. Starting from the top layer, user can access CIRANO using any modern web browser. The requests are handled by Apache Web Servers, used for load balancing and security (`mod_security` plugin). Each Apache Server is connected with a number of Tomcat servlet containers and forwards them the requests, depending on their availability. An application server resides behind each Tomcat and transforms the request to an application functionality query or a database query. Each application server communicates with a set of CIRANO applications and each application has its own database schema in a certain database instance. There are DB proxies in front of all the database servers, offering security and high availability.

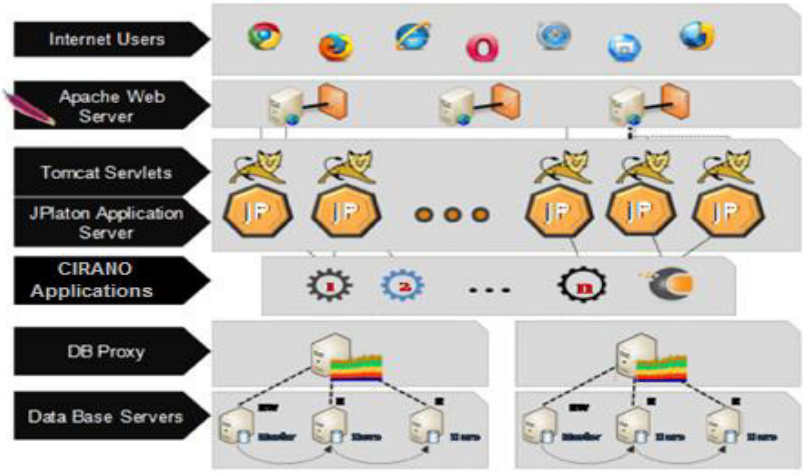


Fig. 3. CPSM architecture of CIRANO

Summarizing, the tools and technologies used to build the CIRANO platform are listed in Table 2.

Table 2. Tools and Technologies used to build the CIRANO Platform.

Tools and Technologies used to build the CIRANO Platform			
Programming Languages	✓ JAVA jdk 1.7 ✓ Android SDK Platform 20 ✓ JAVASCRIPT ✓ HTML 5 ✓ CSS3	Webservers	✓ Tomcat >7 ✓ Apache 2.4
Libraries	✓ jQuery> 1.7 ✓ Google API 1.19 ✓ Ojdbc 6 ✓ Sqljdbc 4.1 ✓ Httpclient 4.3.2 ✓ mysql-connector-java 5.1.25	Database Servers	✓ MySQL 5.6 ✓ Percona 5.6
IDE	✓ Netbeans 8.0 ✓ IntelliJ IDEA – JetBrains 14	Monitoring	✓ Zabbix 2

4. The system in practice

4.1. CIRANO toolboxes and services

CIRANO platform offers the whole spectrum of cloud development services, as depicted in Table 3.

Table 3. CIRANO toolbox and services.

Service / Function	Description
Administration Panel	The administrator can create users and roles, assign roles to users and grant specific access permissions.
Integration with CIRANO jAgora Repository	The developer can log in the CIRANO jAgora repository and download model templates or complete applications and embed them or integrate them in his own application.
Database Designer	Developers can use the graphical interface to create their database (CREATE, ALTER, MODIFY, DELETE) and design their data tables by adding platform specific table field attributes.
Auditing Tool	Using the auditing tool developers have full access to their applications log file. They are able to distinguish between database errors or application errors and identify possible risks or user actions that can be critical for the application
Menu Editor	The Menu Editor toolbox offers the developer the necessary tools to design, code, drag 'n drop menu components, test, edit and build his application's menu
Data Designer	The Data Designer provides a friendly development environment for designing and creating new datasets, queries, lists, variants and data records. The developer can also precompile his design for debugging purposes, while an "auditing" process is available upon saving the designed XML for quality inspection purposes.
E-Object Management	A document repository is available for the users (file sharing, versioning etc).
Device Agent	The platform offers integration with various 3-party devices (hardware peripherals, smartphones, tablets, pebble watch and other hardware devices).
Messenger	A complete communication system is available to the users to communicate with each other during the development process.
Interface Designer	The Interface Designer comes with a handful of design tools, such as forms, tables, panels, HTML areas, bars and more. Components can be added with drag 'n drop and the developer can actually visualize the UI.
Code Editor	The Code Editor tool, lets designers write the event driven procedures in Platonscript. They can also add Javascript or CSS files required by the client side application part.
Repo Publisher	The developer can utilize the Repo Publisher tool in order to publish his application into the CIRANO jAgora marketplace from where it is available for distribution.

CIRANO has already been used and tested during the development of several projects, such as the update of the Comidor collaboration suite[11]. The implementation of the specific project started using JAVA and switched to CIRANO. This transition offered many advantages, which are discussed in the following description of the Comidor case study.

4.2. Comidor collaboration suite

4.2.1. Challenges

Comidor (`https://www.comidor.com`) is a collaboration platform, which offers a plethora of tools that assist organizations in managing their communications, collaboration, content, projects and sales force. As it embeds all these tools into an integrated cloud suite, its development inherited all the difficulties of migrating the implementation to the cloud. In addition, after many years of development, its structure had become obsolete and the continuous functionalities additions had rendered the project difficult to update and monitor. As a result, the development team's productivity had been significantly reduced.

4.2.2. Plan

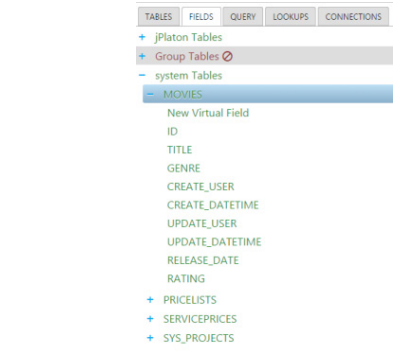
Using the CIRANO platform, Comidor has been restructured following a new layered architecture. After the clear separation of functionalities among the platform's layers, the system consisted of:

- The collaboration system level with basic document, communication, people and task management.
- A project management package layer, which extends some system functionalities, like the task and account management, and adds new features like project monitoring, group task management and resource utilization.
- A CRM package which incorporated all the customer related units, like email campaign management, web analytics and lead tracking.
- A business automation package, which integrates the system layers tasks with the new workflow management processes.
- An accounting package, which was developed from scratch using only CIRANO and added financial tracking and invoicing capabilities.
- Several group levels with categorized functionalities depending on different business sectors (like real estate agencies, accounting offices, vehicle control centers).
- Some user layers which helped on the realization of cloud customizations for different clients.

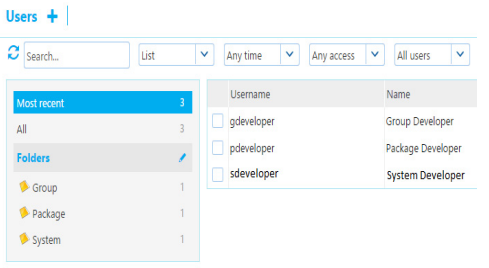
4.2.3. Implementation Steps

More specifically, the update of the existing Comidor system, using the tools of the CIRANO platform, involved the following steps:

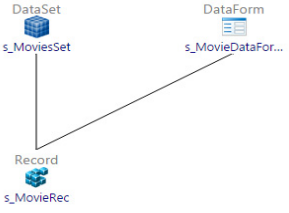
1. The database schema was redesigned using the Database Designer module (Fig.4a). Easy remote database manipulation made the whole development process agile and flexible to changes.
2. Developers were assigned different roles (system, package, group, and user) and the application was separated in small programming units, each of them residing in one of its layers (Fig. 4b).
3. Each team started working only on the features described by the layer's units. Each developer in a team was assigned with a number of programming units to implement.
4. The development of a programming unit was split in data design, business logic coding and user interface design. CIRANO's data modeling tools were proved very effective (Fig. 4c) and data driven procedures provided a simple, clean and easy-to-use coding model (Fig. 4d), while default UI components create a robust interface design (Fig. 4e).
5. Data and code models were reused in many cases, either using the platform's inheritance architecture or by consuming web services, which were built upon those models.



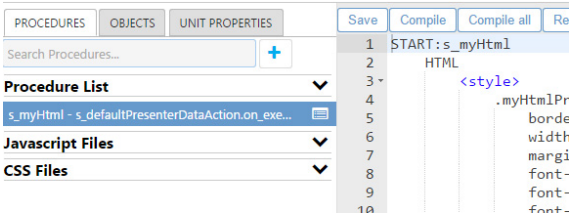
(a)



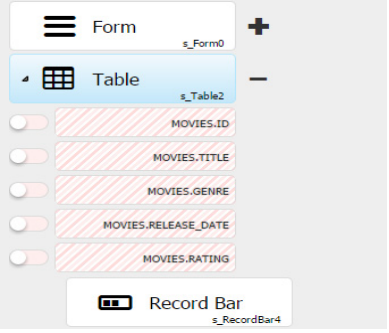
(b)



(c)



(d)



(e)

Figure 4. Utilization of CIRANO tools: In (a) the step 1 (redesign of the database), a part of Cloud Database Designer interface is illustrated. Fig (b) presents step 2, the distribution of roles among development teams, using the Administration Panel. In (c) an inner sight of the Data Designer tool, used in step 4, where sets of data are created and connected with each other, is depicted. Fig. (d) shows a small part of the Code Editor, where code is divided in Platonscript procedures (server side), Javascript and CSS (client side). Finally, in (e) the Interface Designer is illustrated, using a Form, a Table and a Record Bar for data representation.

5. Discussion and Conclusions

In this paper, we presented CIRANO, a novel a cloud IDE for the development of cloud applications. CIRANO's main feature resides in its ability to gather developers in different teams, while it structures the application development in system, package, group and user layers, enabling simultaneous development. Moreover, this structured approach facilitates the development of extensible and easily maintained applications, since each new module can be added as a new layer. Security is addressed by restricting user access only to certain layers. Apart

from the layer restrictions the platform offers also user security per table using access and change rights. Something that could be considered in the future is the ability of a development team to secure whole database tables (and not only layers) so that they are made accessible only within specific layers. According to the initial evaluation of the platform, the provided tools (i.e. Data and Interface Tools) assist the development teams to drastically reduce the time required to design and build applications. Furthermore, real time auditing of both developer and user actions is essential during the debugging and monitoring phase of the development. Creating reports is also a huge overhead for the development teams, due to the scarcity of requirements. CIRANO's reporting tools provide the ability for fast and agile customized reports creation. The overall experience gained from the usage of the platform is that CIRANO enables for agile development in the Cloud era, where rapid response to changing needs, scalability and security are absolutely essential.

It can be argued that developers do not like missing their advanced tools in the variety of programming languages they may have to use in order to build a full cloud application. However, the Cloud itself poses many changes to the already existing languages and frameworks; developers have to adapt to changes. There are already other programming environments, which offer the ability of Cloud code creation and compilation. What CiRANO does, is an attempt to balance the disadvantage of a new programming methodology, with sustainability gains through multi layer architecture, through modeling, MVC and platform given ready tools, so that development in total is far more productive.

Concerning the extension of platform, there are a number of issues that could be examined. A new and advanced version control system closely integrated with GIT, will lead to more effective and error free collaboration. Another aspect that should be examined is the ability to provide hardware–specific resources to each application, through a new virtualization layer. This is the only case where the platform needs to have a notification system in order to inform development teams for possible conflicts created by changes in a certain layer. Last but not least, following the recent trends and the migration of certain types of applications to NoSQL, the platform should be able to maintain and handle connections with such a database. Since there is a jdbc for NoSQL databases, it is a matter of time to develop data platform components to handle this kind of data.

Finally, the E-Object Management module, which offers a built–in document management tool, could be redesigned, in order to support multiple file storage solutions and a desktop client for file synchronization. Further decoupling the file system and the E-Object module, will lead to easier integration with a variety of storage solutions, such as Dropbox (`https://www.dropbox.com`), Google Drive (`drive.google.com`), Gluster (`www.gluster.org`), etc., each one with different special characteristics in respect to confidentiality, integrity and availability.

Acknowledgment

The work reported here has been carried out in the framework of national project Cloud IDE for JPlaton Open Multi-layered Applications (CiRANO), led by B-Open S.A., conducted in the context of the Programme for the Development of Industrial Research and Technology (PAVET) 2013-2015 (code 364-BET-2013) funded by the General Secretariat for Research and Technology (GSRT) and co-funded by the European Commission.

References

1. Barroso, Luiz André, Jimmy Clidaras, and Urs Hölzle. "The datacenter as a computer: An introduction to the design of warehouse-scale machines." Synthesis lectures on computer architecture 8.3 (2013): 1-154.
2. Armbrust, Michael, et al. "A view of cloud computing." *Communications of the ACM* 53.4 (2010): 50-58.
3. Kats, Lennart CL, et al. "Software development environments on the web: a research agenda." Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software. ACM, 2012.
4. Beydeda, Sami, and Matthias Book. Model-driven software development. Vol. 15. Heidelberg: Springer, 2005.
5. Stahl, Thomas, Markus Voelter, and Krzysztof Czarnecki. Model-driven software development: technology, engineering, management. John Wiley & Sons, 2006.

6. Kleppe, Anneke G., et al. "The model driven architecture: practice and promise." (2003).
7. Fowler, Martin. UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional, 2004.
8. Herbst, Nikolas Roman, Samuel Kounev, and Ralf Reussner. "Elasticity in Cloud Computing: What It Is, and What It Is Not." ICAC. 2013.
9. Wu, Jian, Qianhui Liang, and Elisa Bertino. "Improving scalability of software cloud for composite web services." Cloud Computing, 2009. CLOUD'09. IEEE International Conference on. IEEE, 2009.
10. Jenkins, Jam, et al. "JavaWIDE: innovation in an online IDE." Journal of Computing Sciences in Colleges 25.4 (2010): 6-6.